1985-09

An implementation of the projective algorithm for linear programming

Bretschneider, Guenter W.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN IMPLEMENTATION
OF THE
PROJECTIVE ALGORITHM FOR LINEAR PROGRAMMING

by

Guenter W. Bretschneider

September 1985

Thesis Advisor: R. Kevin Wood

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: An Implementation of the Projective Algorithm for Linear Programming	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, September, 1985
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)*: Guenter W. Bretschneider	8. CONTRACT OR GRANT NUMBER*(s)*
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943-5100	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943-5100	12. REPORT DATE: September, 1985
	13. NUMBER OF PAGES: 43
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*
	15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Linear Programming, Projection Method, Symmetric Matrices, Least Squares Problem, Cholesky Factorization

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

An algorithm to solve linear programming problems is presented which is based on Karmarkar's projective method. The algorithm includes a practical method to project a general linear programming problem onto a unit simplex and eliminates the a priori need to know the optimal value of the objective function. The implementation conserves sparsity. The key part of the implementation is the solution of a linear least-squares problem to find an

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. improving direction: a direct and an iterative method are implemented to solve this problem. The direct method employs the minimum-degree heuristic to reorder the system of normal equations, and thus conserve sparsity during the following Cholesky factor of the normal equation martrix as a preconditioner for conjugate gradient iterations which are performed implicitly on the preconditioned matrix. The study concludes with implementation remarks, and computational results.

An Implementation of the Projective Algorithm for Linear Programming

by

Guenter W. Bretschneider
Captain, German Air Force
M.S. in Aeronautical Engineering,
Hochschule der Bundeswehr, Munich, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

ABSTRACT

An algorithm to solve linear programming problems is presented which is based on Karmarkar's projective method. The algorithm includes a practical method to project a general linear programming problem onto a unit simplex and eliminates the *a priori* need to know the optimal value of the objective function. The implementation conserves sparsity. The key part of the implementation is the solution of a linear least-squares problem to find an improving direction: a direct and an iterative method are implemented to solve this problem. The direct method employs the minimum-degree heuristic to reorder the system of normal equations, and thus conserve sparsity during the following Cholesky factorization. The iterative method uses the incomplete Cholesky factor of the normal equation matrix as a preconditioner for conjugate gradient iterations which are performed implicitly on the preconditioned matrix. The study concludes with implementation remarks, and computational results.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

In recent papers Karmarkar [Refs. 1,2] has presented a new method for the solution of linear programming (LP) problems: His new solution technique moves from some feasible starting point across the interior region of a polytope that is defined by the problem constraints. He shows that the number of steps to find an optimal solution with his technique is polynomially bounded.

In contrast, the simplex algorithm which, is widely used for the solution of LP problems finds an optimal solution by moving from vertex to vertex on the polytope. This is known, in the worst case, to require an exponential number of steps [Ref. 3].

The polynomial bound of the projective algorithm makes the new solution technique very appealing to researchers. The theory of the new algorithm seems to be widely accepted among experts, while Karmarkar's claim that his algorithm is 50-100 times faster than the simplex method has met skepticism [Ref. 4].

In this study a variant of the projective method is implemented, and some well known test problems are solved.

A. THE BASIC PROJECTIVE METHOD

Following Karmarkar [Ref. 1: p. 4], the number of steps of the algorithm depends on R/r, where R is the radius of the sphere circumscribing the polytope, and r the radius of the inscribed sphere.

Assume a general LP of the form

$$\begin{aligned} \text{Min} \quad & c^T x \\ \text{s.t.} \quad & A x = b \\ & x \geq 0 . \end{aligned} \qquad \text{(LP1)}$$

With the assumption that the sum of its variables has an upper bound, and with the proper scaling of variables, a convexity constraint

$$1^T x = 1 \tag{1.1}$$

can be added to LP1. This transformation maps the LP onto a unit simplex S whose center is at $a_o = (1/n,\ldots,1/n)$. Then, a transformation is performed such that a feasible interior starting point is mapped onto a_o.

If $B(a_o,r)$ is the largest sphere with center a_o that can be inscribed into the simplex S, and $B(a_o,R)$ is the smallest sphere circumscribing S, then

$$R/r = n. \tag{1.2}$$

By restricting a solution x to remain inside the largest inscribed sphere $B(a_o,r)$, the method achieves in one iteration a reduction in the difference between the current objective value and the optimal objective value by a factor of $(1 - 1/n)$. Following Shanno [Ref. 5] a simple proof is: Let

$$f^* = \min c^T x, \quad x \in S, \; Ax = b, \tag{1.3}$$

$$\underline{f} = \min c^T x, \quad x \in B(a_o,r), \; Ax = b, \; 1^T x = 1, \tag{1.4}$$

$$\overline{f} = \min c^T x, \quad x \in B(a_o,R), \; Ax = b, \; 1^T x = 1. \tag{1.5}$$

Then,

$$c^T a_o - \underline{f} \leq c^T a_o - f^* \leq c^T a_o - \overline{f}, \tag{1.6}$$

and with equation 1.2

$$c^T a_o - \overline{f} = n(c^T a_o - \underline{f}). \tag{1.7}$$

From that

$$(\underline{f} - f^*)/(c^T a_o - f^*) \leq (1 - 1/n). \tag{1.8}$$

If a linear objective function could be maintained at each iteration, it follows that an upper bound on the number of steps required to find an optimal solution is of O(n). Unfortunately, the projective transformations needed to continue the algorithm result in a nonlinear objective function.

B. DERIVATION OF THE ALGORITHM

1. *The Canonical Form*

Suppose we have a linear programming problem of the form LP1. Karmarkar's method requires that this LP be transformed into the following canonical form,

$$\begin{aligned} \text{Min} \quad & c^T x \\ \text{s.t.} \quad & A x = 0 \\ & 1^T x = 1 \\ & x \geq 0, \end{aligned} \qquad \text{(LP2)}$$

where the optimal solution value is zero. With the assumption that the sum of the variables of an LP is bounded above and subsequent scaling by this bound, a convexity constraint (equation 1.1) can be added to LP1. In practice this can be a problem because choosing too large an upper bound may cause numerical problems.

LP2 requires that the nonhomogeneous system of equations $Ax = b$ be made homogeneous. Karmarkar [Ref. 1: p.34] proposes a transformation that would transform the i-th equation $a_i^T x = b_i$ to

$$\sum_j (a_{ij} - b_i) x_j = 0. \qquad (1.9)$$

This transformation has the disadvantage that when b is dense the sparsity of A will be lost. Karmarkar requires the optimal solution value to LP2 to be zero together with a special stopping criterion (equation 1.26), to prove the polynomial bound. To achieve the zero objective value the optimal objective function value f^* of the

untransformed LP has to be known in advance, and the following transformation made

$$c^T x - f^* = c^T x - f^* 1^T x = (c - f^* 1)^T x \ . \qquad (1.10)$$

Karmarkar concludes [Ref. 2: p. 387] that if the minimal objective value determined by the projective algorithm is not equal to zero, the original problem must be either infeasible or unbounded. Another method for making the transformation from LP1 to LP2 is discussed in Chapter II.B.

2. *The Projective Transformation*

Let $x^o > 0$ be a known feasible solution to LP2. The invertible projective transformation

$$y = \frac{D^{-1}x}{1^T D^{-1} x} \qquad (1.11)$$

maps any x, such that $1^T x = 1$ and $x \geq 0$, onto y such that $1^T y = 1$ and $y \geq 0$. D is a diagonal matrix whose entries are $(x_1^o, \ldots, x_n^o)$.

The transformation maps the LP2 unit simplex in x-space onto another unit simplex in y-space. The point x^o is mapped onto $y^o = 1/n\ 1^T$, the center of the unit simplex in y-space. The inverse of the transformation (equation 1.11) is given by

$$x = \frac{D\ y}{1^T D\ y} \ . \qquad (1.12)$$

After the transformation, LP2 can now be restated as

$$\text{Min} \quad \frac{c^T D\, y}{1^T D\, y}$$

$$\begin{aligned} \text{s.t.} \quad A\, D\, y &= 0 \\ 1^T\, y &= 1 \\ y &\geq 0 \,. \end{aligned} \qquad \text{(LP3)}$$

Strictly speaking LP3 is not a linear program because its objective function is a rational function of y. Karmarkar [Ref. 1: page 18ff] shows that it is sufficient to consider a linear approximation to the objective function of LP3, which leads to

$$\begin{aligned} \text{Min} \quad & c^T D\, y \\ \text{s.t.} \quad A\, D\, y &= 0 \\ 1^T\, y &= 1 \\ y &\geq 0 \,. \end{aligned} \qquad \text{(LP4)}$$

3. *Optimization Over a Sphere*

A solution to LP4 is now restricted to lie within a sphere with center at y^0 and radius αr, where

$$r = 1/(n(n-1))^{-1/2} \,. \qquad (1.13)$$

r is the radius of the largest sphere that can be inscribed into the unit simplex, and α is a constant such that $0<\alpha<1$. α provides a margin which ensures that the algorithm doesn't select a point outside the sphere due to round-off error. By convexity, an optimal solution will occur at the boundary of the sphere. Thus, the additional constraint can be stated as an equality. Now we have the following version of the LP:

$$
\begin{aligned}
\text{Min} \quad & c^T D\, y & & \\
\text{s.t.} \quad & A\, D\, y & = & \ 0 \\
& 1^T y & = & \ 1 \\
& (y-y^o)^T (y-y^o) & = & \ \alpha^2 r^2 \ .
\end{aligned}
\qquad \text{(LP5)}
$$

Before the problem can be solved, one more transformation that moves the center of the sphere to the origin is useful. Let $y = \bar{y} + y^o$, and eliminate the constant terms $ADy^o = 0$, $1^T y^o = 1$ and $c^T D y^o$. For convenience, define B by

$$
B = \begin{bmatrix} A\ D \\ \hline 1^T \end{bmatrix} \qquad (1.14)
$$

and the gradient $\bar{c}$ of the objective function of LP5 by

$$
\bar{c} = c^T D \ . \qquad (1.15)
$$

Then LP5 can be restated as,

$$
\begin{aligned}
\text{Min} \quad & \bar{c}^T \bar{y} & & \\
\text{s.t.} \quad & B\, \bar{y} & = & \ 0 \\
& \bar{y}^T \bar{y} & = & \ \alpha^2 r^2 \ .
\end{aligned}
\qquad \text{(LP6)}
$$

In order to solve LP6, note that an initial feasible solution to LP6 is $\bar{y} = 0$ (which corresponds to $y = y^o$ in LP5). This is also the center of the sphere $\bar{y}^T \bar{y} = \alpha^2 r^2$. An optimal solution to LP6 can be obtained by finding a direction of maximum rate of ascent $\hat{c}$ that is feasible with respect to $B\bar{y} = 0$, and moving in direction $-\hat{c}$ (maximum rate of descent) a distance αr from $\bar{y} = 0$ to the boundary of the sphere.

A feasible direction of maximum ascent is found by orthogonally projecting $\bar{c}$ onto the null space of B (see [Ref. 1: page 17]), i.e., the following problem has to be solved in terms of c_p

$$\min_{c_p} \ \bar{c}^T\bar{c} - 2\bar{c}^T c_p + c_p{}^T c_p \quad = \quad \min_{c_p} \ (\| \bar{c} - c_p \|_2)^2 \qquad (1.16)$$

$$\text{s.t.} \quad Bc_p \ = \ 0 \ .$$

Define the Lagrangian $L(c_p, \lambda)$ by

$$L(c_p,\lambda) = \bar{c}^T\bar{c} - 2\bar{c}^T c_p + c_p{}^T c_p - \lambda^T B c_p. \qquad (1.17)$$

First-order optimality conditions for the Lagrangian yield

$$-2\bar{c} + 2c_p - B^T\lambda \ = \ 0, \qquad (1.18)$$

and

$$Bc_p \ = \ 0. \qquad (1.19)$$

After multiplying by B and dropping $2Bc_p=0$ (equation 1.19) we get

$$-2B\bar{c} \ = \ BB^T\lambda \ . \qquad (1.20)$$

Assuming $(BB^T)^{-1}$ exists, we can solve for λ

$$\lambda = -2(BB^T)^{-1}B\bar{c}. \qquad (1.21)$$

Substituting equation 1.21 into equation 1.18 gives

$$c_p = \bar{c} - B^T(BB^T)^{-1}B\bar{c}. \qquad (1.22)$$

The direction of maximum rate of ascent is then

$$\hat{c} = c_p / \| c_p \| \ . \qquad (1.23)$$

Thus, the optimal solutions to LP6 and LP5 are

$$\bar{y} = -\alpha r\hat{c}, \qquad (1.24)$$

and

$$y^1 = y^o - \alpha r\hat{c} \tag{1.25}$$

respectively.

Finding c_p is the key part of Karmarkar's method because it involves the major portion of the computational work. Solving equation 1.22 for c_p can be viewed as solving a linear least-squares problem (see Chapter II.C.).

With the optimal solution to LP5 in y-space, the method proceeds by transforming that solution back into x-space by use of equation 1.12. Next it defines a new matrix D and iterates until it reaches the stopping criterion [Ref. 1: p. 14]

$$(c^T x^k)/(c^T x^o) \leq 2^{-q} \tag{1.26}$$

where q is a termination parameter. Note that $c^T x = 0$ at optimality. Table 1 shows an outline of Karmarkar's basic method.

TABLE 1

ALGORITHM 1

Input Problem Size.....n

Coefficient Matrix.....A

Cost Function.....c^T

Initial Feasible Solution...x^o

Termination Parameter.....q

Feasibility Parameter.....α

Begin $k = 1$

$x = x^o$

$f^o = c^T x^o$

$f = \infty$

$r = 1/(n(n-1))^{-1/2}$

While $(f/f^o > 2^{-q})$

$D = \text{diag}(x)$

$y = (D^{-1}x)/(1^T D^{-1}x) = 1$

$\bar{c} = c^T D$

$$B = \begin{bmatrix} A\ D \\ \hline 1^T \end{bmatrix}$$

$c_p = \bar{c} - B^T(BB^T)^{-1}B\bar{c}$

$\hat{c} = c_p/\|c_p\|$

$y = y - \alpha r\hat{c}$

$x = (Dy)/(1^T Dy)$

$f = c^T x$

$k = k + 1$

End(While)

End

Output Solution.....x

Objective Function Value...f

Iteration Count.....k

II. A VARIANT OF THE PROJECTIVE METHOD

A. INTRODUCTION

The following sections outline a variant of the projective method that was proposed by Wood [Ref. 6]. It has a practical method of bringing an LP into the canonical form, and uses the non-linear objective function of LP3 to find a gradient c' which corresponds to c of equation 1.16. Similar to the simplex method, the proposed algorithm uses a ratio test to determine a feasible step length.

Other practical features of the proposed method include the relaxation of the requirement to know the optimal objective value in advance, and exploitation of sparsity of A. The implementation is especially concerned with controlling fill-in during the solution of the normal equations.

B. DERIVATION OF THE MODIFIED ALGORITHM

Given an LP problem of the form LP1, we use a single artificial variable x_{n+1} to attain initial feasibility:

$$\begin{array}{lll} \text{Min} & (c^T, M)(x, x_{n+1}) & \\ \text{s.t.} & Ax-(Ax^o-b)x_{n+1} = b & \text{(LP7)} \\ & x, x_{n+1} \geq 0 \ . & \end{array}$$

where M is the cost for the artificial variable and x^o is the initial solution with

$$(x^o, x^o{}_{n+1}) = 1^T \ . \qquad (2.1)$$

The transformation has added one more column to the problem.

To get a homogeneous right-hand side in LP7 we introduce an additional variable, and apply the following projective transformation, whose inverse is given by

$$(x,x_{n+1}) = (n+2)(x',x'_{n+1})/x'_{n+2}, \tag{2.2}$$

$$(x',x'_{n+1}) = (n+2)(x,x_{n+1})/(1^T(x,x_{n+1})+1) \tag{2.3}$$

$$x'_{n+2} = (n+2)/(1^T(x,x_{n+1})+1) \ . \tag{2.4}$$

Let the artificial column be denoted by a, i.e. $a = -(Ax^o-b)$. The transformed problem can now be restated, with the exception of the objective function, in Karmarkar's canonical form,

$$\begin{array}{lll} \text{Min} & \dfrac{(c^T,M,0)(x',x'_{n+1},x'_{n+2})}{x'_{n+2}} & \\ \text{s.t.} & (A,a,-b)(x',x'_{n+1},x'_{n+2}) = 0 & \text{(LP8)} \\ & 1^T(x',x'_{n+1},x'_{n+2}) = n+2 & \\ & (x',x'_{n+1},x'_{n+2}) \geq 0. & \end{array}$$

The above transformation adds yet another column to the problem, but has the advantage that it doesn't change the sparsity of A, as opposed to Karmarkar's proposal. Rather than projecting the LP problem onto a unit simplex, it is projected onto an (n+2)-simplex. This is done to improve numerical stability.

The projective transformation, equation 1.12, is applied to LP8 which gives

$$\begin{array}{lll} \text{Min} & (c^T,M,0)D(y,y_{n+1},y_{n+2})/d_{n+2}y_{n+2} & \\ \text{s.t.} & (A,a,-b)D(y,y_{n+1},y_{n+2}) = 0 & \text{(LP9)} \\ & 1^T(y,y_{n+1},y_{n+2}) = n+2 & \\ & (y,y_{n+1},y_{n+2}) \geq 0 \ . & \end{array}$$

The gradient (compare with equation 1.16) of the objective function of LP9, evaluated at the initial feasible starting point $y^o = 1^T$ is proportional to

$$c' = (c_1 d_1, \ldots, c_n d_n, M d_{n+1}, -c^T x). \tag{2.5}$$

The step of normalizing c_p in Algorithm 1 is replaced by a ratio test. Let

$$jmax = \text{argmax}\{(c_p)_j\}. \tag{2.6}$$

This allows the algorithm to make a step outside the inscribed sphere, but maintains feasibility by restricting a solution to lie inside the simplex. Then, the update of y becomes

$$y = 1 - \rho c_p / (c_p)_{jmax} , \tag{2.7}$$

where ρ is a parameter to maintain feasibility such that $0<\rho<1$. Table 2 shows the modified algorithm.

C. THE LINEAR LEAST-SQUARES PROBLEM

Computation of the projected gradient c_p during every iteration accounts for most of the computational workload in any algorithm based on Karmarkar's method. Solving equation 1.22 can be viewed as solving the following linear least-squares problem,

$$\text{Min} \quad (\| c' - B^T \hat{\lambda} \|_2)^2 \tag{2.8}$$

where B^T is an $(n+2)$xm matrix with $m \leq (n+2)$ assumed.

If $\text{rank}(B^T)=m$, then the solution to (2.8) is given by the solution to the system of normal equations

$$BB^T \hat{\lambda} = Bc' . \tag{2.9}$$

The projected gradient c_p is then the residual vector of the least-squares problem (2.8), i.e.,

$$c_p = c' - B^T \hat{\lambda} . \tag{2.10}$$

TABLE 2

ALGORITHM 2

Input Problem Size.....n

Coefficient Matrix.....A

Cost Function.....c^T

Initial Feasible Solution...x^o

Termination Parameter.....t

Feasibility Parameter.....ρ

Begin

$k = 1$

$x = x^o$

$f^o = \infty$

While $(f^o - f(x) > t)$

$D = \text{diag}(x)$

$f^o = f(x)$

$y = (n+2)(D^{-1}x)/(1^T D^{-1}x)$

$c' = \nabla_y f(y)$

$B = \begin{bmatrix} A\ D \\ \hline 1^T \end{bmatrix}$

$c_p = c' - B^T(BB^T)^{-1}Bc'$

$j_{max} = \text{argmax}(c_p)_j$

$y = 1 - \rho c_p/(c_p)_{jmax}$

$x = (n+2)(Dy)/(1^T Dy)$

$k = k + 1$

End(While)

End

Output Solution.....x

Objective Function Value...f

Iteration Count.....k

BB^T is symmetric and positive definite, given that B^T has full column rank.

As Heath [Ref. 7: p. 499] points out, the ideal choice for solving the normal equations, given full rank, is Cholesky factorization. If B^T is not of full column rank the cross-product matrix will be singular and $(BB^T)^{-1}$ ceases to exist. BB^T could become nearly singular if B^T is near rank degenerate. Shanno [Ref. 5: p. 25] shows that this will happen if the optimal solution is degenerate, i.e. as the optimum is approached numerical problems arise and Cholesky factorization is likely to fail.

One problem that cannot be avoided when solving the normal equations is the fact that the P-condition number of BB^T is the square of that of B^T [Ref. 8: p. 223], so that when B^T is already ill-conditioned it may be impossible to find an accurate solution to equation 2.9.

Another important consideration when computing BB^T is that the sparsity in B^T will not automatically guarantee sparsity in BB^T. In fact, the addition of variables in LP7 and LP8 has added two possibly dense bottom rows into B^T. Thus, BB^T will be completely dense. However, one can cope with that by initially omitting the dense rows in B^T from the computation of BB^T, and later updating the solution to equation 2.9 using procedures similar to the ones described in [Ref. 9: p. 58-65].

D. IMPLEMENTATION

Algorithm 2 has been implemented in FORTRAN H (Extended) Opt(2) on an IBM 3033 AP under VM/CMS. All floating point arithmetic is performed in double precision. The program is designed to accept different solution modules from available software packages for solving the linear least-squares problem.

Input data sets are in standard MPS format. The numerical values of the non-zero elements of the constraint matrix A are stored column-wise in a real array. For versatility a full set of pointers are defined:

1. IC column index
2. R row index

3. AP location of first non-zero element in a column
4. RP location of last non-zero element in a row
5. LINK location of next non-zero element (backwards) in a row.

Brameller, Allan and Hamam [Ref. 10: p. 104-110] give a comprehensive discussion of sparse storage schemes.

The transformation of an LP problem into the canonical form adds one dense row and two dense columns to the A matrix. The dense row originates from the convexity constraint equation 1.1, the first dense column stems from the single artificial variable that is needed to attain initial feasibility, and the second dense column is added to make the system of equations homogeneous (see LP8). The artificial column is updated with the residual of the current solution as long as the total infeasibility is above a specified threshold.

As mentioned earlier, dense rows in B^T yield BB^T completely dense. With the following method, this problem can be alleviated. The method applies to any number of dense rows in B^T, but in this study we are only concerned about two dense rows, namely the ones that result from the transformations that are performed to get from LP2, to LP8 via LP7. Consider the projection problem of equation 1.16 in the following form,

$$\text{Min} \quad (\|c' - c_p\|_2)^2 \qquad (2.11)$$
$$\text{s.t.} \quad Bc_p = 0$$

Replace c_p by z, and let

$$B = (B_1, B_2) \qquad (2.12)$$

$$c' = (c'_1, c'_2) \qquad (2.13)$$

$$z = (z_1, z_2)^T \qquad (2.14)$$

where B_1 is an (mxn_1), B_2 an (mxn_2) matrix, and n_2 is the number of dense columns in B. Equation 2.11 now becomes

$$\begin{aligned} &\text{Min} \quad (\|c'_2 - z_2\|_2)^2 + (\|c'_1 - z_1\|_2)^2 \\ &\text{s.t.} \quad B_1 z_1 = -B_2 z_2 \end{aligned} \tag{2.15}$$

and when separated, 2.15 becomes

$$\underset{z_2}{\text{Min}} \left\{ (\|c'_2 - z_2\|_2)^2 + \underset{z_1}{\text{Min}} \; (\|c'_1 - z_1\|_2)^2 \right. \qquad \text{s.t. } B_1 z_1 = -B_2 z_2 \; . \tag{2.16}$$

Solving the inner minimization first, the Kuhn-Tucker conditions yield

$$c'_1 - z_1 = B_1{}^T \lambda \; , \tag{2.17}$$

and

$$B_1 z_1 = -B_2 z_2 \; . \tag{2.18}$$

Multiplying 2.17 with B_1 we get,

$$B_1 c'_1 - B_1 z_1 = B_1 B_1{}^T \lambda \; . \tag{2.19}$$

Substituting 2.18 into 2.19 gives

$$B_1 c'_1 + B_2 z_2 = B_1 B_1{}^T \lambda \; . \tag{2.20}$$

Let λ_0 solve

$$B_1 c'_1 = B_1 B_1{}^T \lambda \tag{2.21}$$

and let λ_j solve

$$(B_2)_j = B_1 B_1{}^T \lambda \; , \; j = 1, \ldots, n_2 \tag{2.22}$$

where $(B_2)_j$ is the j-th column of B_2, then 2.22 corresponds to solving the matrix equation system

$$B_2 = B_1 B_1^T \underline{\lambda} \tag{2.23}$$

where $\underline{\lambda}$ is an (mxn_2) matrix. Then, the solution to λ for the inner minimization is,

$$\lambda = \lambda_0 + \underline{\lambda} z_2 \ . \tag{2.24}$$

Substituting 2.24 into 2.17 gives

$$c'_1 - z_1 = B_1^T \lambda_0 + B_1^T \underline{\lambda} z_2 \ . \tag{2.25}$$

To simplify notation, let $h = B_1^T \lambda_0$ and $H = B_1^T \underline{\lambda}$

Then equation 2.25 becomes

$$z_1 = c'_1 - h - Hz_2 \ . \tag{2.26}$$

Substituting 2.26 into 2.15, the overall minimization then becomes

$$\underset{z_2}{\text{Min}} \ (\| c'_2 - z_2 \|_2)^2 + (\| h + Hz_2 \|_2)^2 \tag{2.27}$$

which is a simple least-squares problem. The first-order Kuhn-Tucker optimality conditions yield,

$$c'_2 - h^T H = (I + H^T H) z_2 \tag{2.28}$$

which gives, solving for z_2,

$$z_2 = (I + H^T H)^{-1} (c'_2 - h^T H) \ . \tag{2.29}$$

With the solutions to 2.29 and 2.26 we have the desired result,

$$c_p = z = (z_1, z_2)^T \ . \tag{2.30}$$

In practice the following procedure is followed:

1. compute $B_1B_1^T$
2. factor $B_1B_1^T$, e.g. using Cholesky factorization
3. solve $B_1B_1^T\lambda = B_1^Tc'_1$, the solution is λ_o
4. compute $h = B_1^T\lambda_o$
5. solve $B_1B_1^T\lambda = (B_2)_1$, the solution is $\underline{\lambda}_1$
6. solve $B_1B_1^T\lambda = (B_2)_2$, the solution is $\underline{\lambda}_2$
7. compute $H = B_1^T\underline{\lambda}$
8. compute $(I + H^TH)^{-1}$, (note that H^TH is a 2x2 matrix if there are 2 dense rows in B^T)
9. compute $(c'_2 - h^TH)$
10. compute z_2
11. compute z_1

The given procedure is efficient since the factorization of BB^T is computed only once and the same system is solved three times using different right hand-sides each time.

As a stopping criterion for the algorithm the following rule is used,

$$\text{IF} \quad \underset{j}{\text{argmax}} \; (|x_j^k - x_j^{k-1}|) \leq t \quad \text{STOP} , \tag{2.31}$$

where t is a real constant. The convergence criterion

$$(\| c_p \| / |c^Tx^o|) < t \tag{2.32}$$

mentioned by Lustig [Ref. 4: p. 12] can also be used to terminate the algorithm.

III. SOLVING THE LINEAR LEAST-SQUARES PROBLEM

A. INTRODUCTION

In this study, the linear least-squares problem is solved by explicitly computing and solving the normal equations. Although the normal equation approach experiences problems when applied to ill-conditioned or near rank deficient matrices, it behaves acceptably with sparse and well-conditioned matrices [Ref. 11].

The numerical methods for solving normal equations generally fall into two classes, direct and iterative methods. One representative of each class is considered in this study. Little can be said as to which class of methods is better, except that direct methods are more attractive in terms of computational work, and iterative methods may require less storage [Ref. 12: p. 11].

B. CHOLESKY FACTORIZATION

The method implemented is given in George and Liu [Ref. 12], and uses Cholesky factorization with a minimum-degree ordering to solve a large sparse positive definite system of equations. Since BB^T is only guaranteed to be positive semidefinite, a modification to the Cholesky factorization algorithm is considered in Chapter IV.A.3. to accommodate the semidefinite case.

The minimum-degree algorithm is a reordering heuristic which attempts to reduce fill-in during the factorization phase. The reordering phase is entirely symbolic; it amounts to a symmetric row and column permutation of BB^T which corresponds to reordering the columns in B^T. During this phase, BB^T doesn't have to be computed numerically; only its structure has to determined. Also, the factorization is first performed symbolically, thus allowing a static data structure for the Cholesky factor L. An outline of the phases of the algorithm is given in Table 3. See also Heath [Ref. 7: p. 499].

TABLE 3

MINIMUM-DEGREE ORDERING / CHOLESKY FACTORIZATION ALGORITHM (MDOC)

1. Determine the nonzero structure of BB^T.
2. Find a permutation matrix P such that PBB^TP^T has a sparse lower triangular Cholesky factor L.
3. Factor PBB^TP^T symbolically and set up the data structure for L.
4. Compute PBB^TP^T numerically.
5. Factor $PBB^TP^T = LL^T$ numerically.
6. Solve $Lz = PBb'$ (back substitution).
7. Solve $L^Ty = z$ (forward substitution).
8. $x = P^Ty$.

1. *The Minimum-Degree Ordering Heuristic*

The heuristic finds an ordering of a symmetric matrix such that fill-in is low when the matrix is being factored. The basic idea is, at each (simulated) factorization step, to permute the part of BB^T remaining to be factored so that a column with the fewest nonzeros is in the pivot position. The implementation consists of six subroutines that are given in George and Liu [Ref. 12: pp. 124-137]. The subroutines accept as input the adjacency graph associated with BB^T represented by an adjacency structure, and return as output a symmetric permutation of BB^T given as a permutation vector for the columns of B^T.

Let $G=(X,E)$ be the adjacency graph of BB^T, where X is the set of nodes, and E is the set of edges. Then, the nodes correspond to the variables of the least-squares problem, i.e. the columns of B^T. Two nodes x and y are said to be adjacent if $\{x,y\}$ is an edge in E. The adjacent set of Y, $Y \subset X$ is defined and denoted by

$$Adj(Y) = (x \in X\text{-}Y \mid \{x,y\} \in E \text{ for some } y \in Y). \qquad (3.1)$$

An adjacency list for $x \in X$ is a list of all nodes in $Adj(\{x\})$. Finally, an adjacency structure for the graph G is the set of adjacency lists for all $x \in X$ [Ref. 12: pp. 37-41]. The particular adjacency structure used in the ordering heuristic stores elements in each adjacency list in contiguous locations. An entry point array to the first element in each list allows access to the list.

The ordering heuristic is based on graph theory, and involves the notion of elimination graphs, quotient graphs, reachable sets and indistinguishable nodes. George and Liu [Ref. 12: pp. 92-124] may be consulted for more details.

2. *Factorization and Solution*

The components of the lower triangular Cholesky factor L of BB^T are computed using the so called "inner product form" algorithm [Ref. 12: p. 20]. The elements of L are given by,

$$l_{jj} = (e_{jj} - \sum_{k=1}^{j-1} l^2_{jk})^{1/2} \qquad \text{for } j=1,2,\ldots,m \qquad (3.2)$$

$$l_{ij} = (e_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk})/l_{jj} \qquad \text{for } i=j+1,j+2,\ldots,m \qquad (3.3)$$

where the e_{ij} are the elements of BB^T.

After the factorization has been computed, the following two linear systems have to be solved (see also Table 3):

$$Lz = Pb' , \qquad (3.4)$$

and

$$L^T y = z . \qquad (3.5)$$

Solving system 3.4 by back substitution involves the use of "inner products" [Ref. 12: p. 25], defined by

$$z_i = (b'_i - \sum_{k=1}^{i-1} l_{ik}x_k)/l_{ii} \qquad \text{for } i=1,2,\ldots,m \ , \tag{3.6}$$

where b'_i stands for the i-th element of the right-hand side of (3.4). For this case L must be accessed row-by-row.

System 3.5 is solved by forward substitution using "outer products" [Ref. 12: p. 26], defined by

$$y_i = z_i/l_{ii} \qquad \text{for } i=1,2,\ldots,m \tag{3.7}$$
$$(z_{i+1},..,z_m) \leftarrow (z_{i+1},..,z_m) - y_i(l_{i+1,i},..,l_{m,i}) \ .$$

For the latter case, L^T must be accessed row-by-row, or L column-by-column instead.

C. INCOMPLETE CHOLESKY FACTORIZATION

This method is an implementation by Ajiz and Jennings [Ref. 13] of the incomplete Cholesky conjugate gradient algorithm (ICCG), whose theory is given in [Refs. 14,15]. The algorithm requires that the coefficient matrix of a set of simultaneous linear equations be symmetric and positive definite. It consists of two distinct parts, one being the Cholesky factorization, which can be complete or incomplete, and the other a conjugate gradient iteration to solve a preconditioned linear system, where the Cholesky factor serves as the preconditioner.

Golub and Van Loan [Ref. 16: pp. 373-377] point out that preconditioning is essential for obtaining good convergence rates with conjugate gradient methods. The convergence rate is closely linked to the P-condition number, which is the ratio of the largest to the smallest eigenvalue. It was mentioned earlier that the condition number of BB^T will be the square of the one of B^T. Ill-conditioned problems have large condition numbers, and hence slow convergence. Preconditioning is a process of transforming a linear system so that its P-condition number is improved [Ref. 17: p. 979].

Consider again the original linear least-squares problem (2.9) with a generic right-hand side b' and x the unknowns,

$$BB^T x = b'. \tag{3.8}$$

Then, equation 3.8 can be preconditioned with a transformation matrix L giving

$$L^{-1}BB^T L^{-T} x = L^{-1}b', \tag{3.9}$$

or

$$L^{-1}BB^T L^{-T} y = b, \tag{3.10}$$

where $y = L^T x$ and $\bar{b} = L^{-1}b'$. According to Ajiz and Jennings [Ref. 13: p. 950] the ideal choice of transformation matrix L is the Cholesky factor of BB^T, since

$$L^{-1}BB^T L^{-T} = I\ , \tag{3.11}$$

provided one could perform exact arithmetic.

The objectives of the incomplete factorization phase of the algorithm are to transform BB^T as close as possible to I, and to reduce fill-in in the factor L. This is accomplished by discarding some off-diagonal coefficients during the factorization, whose magnitudes fall below a preset threshold limit. The result of this operation is an incomplete Cholesky factor L that must satisfy

$$BB^T = LL^T - C\ , \tag{3.12}$$

where C is the matrix of elements omitted from the factorization. Unfortunately, omission of elements from the factorization process can destroy the positive definiteness property and hence lead to a breakdown of the process. Ajiz and Jennings [Ref. 13: pp. 950-951] have

shown that introducing diagonal modifications in C will retain the positive definiteness property. The matrix C will be symmetric and will have diagonal elements that are greater than or equal to zero. Thus, C is a positive semidefinite matrix. Assuming BB^T to be positive definite, adding C will result in LL^T also being positive definite.

Convergence of conjugate gradient iterations is only guaranteed for the positive definite case. Thus, a modification to adapt the Cholesky factorization to the positive semidefinite case as with the direct method may cause slow convergence. The modification is considered in Section IV.A.3. Table 4 gives an outline of the ICCG algorithm.

TABLE 4

INCOMPLETE CHOLESKY - CONJUGATE GRADIENT (ICCG) ALGORITHM

1. Obtain L, an incomplete Cholesky factor of BB^T.
2. Solve $L\bar{b} = b'$ for $\bar{b}$ by forward substitution.
3. Solve $L^{-1}BB^TL^{-T}y = \bar{b}$ for y by conjugate gradient iteration.
4. Determine x by back substitution in $L^Tx = y$.

1. *The Incomplete Factorization*

The procedure presented here is given in Ajiz and Jennings [Ref. 13: pp. 951-952], and Jennings and Malik [Ref. 14: pp. 310-313]. To see how the elements in column j of L are computed consider the following. From matrix equation 3.12 a typical elemental equation may be written as

$$l_{jj}l_{ij} = e_{ij} + c_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk} \qquad (i \geq j) \qquad (3.13)$$

where the subscripts for elements l refer to their positions in the lower triangular factor L, and the e_{ij} are the elements of BB^T. Let

$$e^*_{ij} = e_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk} \quad . \tag{3.14}$$

Then, assuming the elements of L have been computed for columns 1 to (j-1), all the e^*_{ij} in column j can be computed. First consider the case where all the e^*_{ij} pass the rejection test, i.e., are to be retained. Hence, by setting $c_{ij}=0$ in equation 3.13, we get

$$l_{ij} = e^*_{ij}/l_{jj} \quad . \tag{3.15}$$

In case any e^*_{ij} is to be rejected, l_{ij} is set to zero, implying $c_{ij}=-e^*_{ij}$. The off-diagonal term c_{ij} implies diagonal additions c_{jj} and c_{ii} to the matrix C in order to have the positive semidefinitness property. The matrix C doesn't have to be stored in memory; only the diagonal additions c_{jj} are of further interest.

Before any l_{ij} can be computed, l_{jj} has to be determined. With all the diagonal additions c_{jj} that resulted from rejections of e^*_{ij} during the computation of columns 1 to (j-1), an expression for l_{jj} becomes

$$l_{jj} = (e^*_{jj} + \sum_{k=j+1}^{m} c_{jj}(k))^{1/2} \tag{3.16}$$

where e^*_{jj} is defined by

$$e^*_{jj} = e_{jj} + \sum_{k=1}^{j-1} c_{jj}(k) - \sum_{k=1}^{j-1} l_{ik} \quad , \tag{3.17}$$

and $c_{jj}(k)$ is the diagonal addition to c_{jj} resulting from deletion of e^*_{kj}, and $c_{ii}(k)$ the diagonal addition to c_{ii} resulting from deletion of e^*_{ik}.

The rejection operation tests the magnitude of an element e^*_{ij} in relation to the current values of the corresponding diagonal elements $\bar{e}_{jj}$ and $\bar{e}_{ii}$ respectively, whose values are given by

$$\bar{e}_{jj} = e^*_{jj} + \sum_{k=j+1}^{i-1} c_{jj}(k) \quad , \tag{3.18}$$

where the index k refers to the rows for which rejections in column j have had a bearing on $\bar{e}_{jj}$, and

$$\bar{e}_{ii} = e_{ii} + \sum_{K=1}^{j-1} c_{ii}^{(k)} \quad . \tag{3.19}$$

In equation 3.19, k refers to the columns for which rejections in row i have had a bearing on $\bar{e}_{ii}$. An element e^*_{ij} is rejected if

$$e^{*}_{ij}{}^2 < \psi^2 \, \bar{e}_{jj}\bar{e}_{ii} \quad , \tag{3.20}$$

where ψ is the preset rejection parameter. A choice of ψ=0 will retain all elements, thus leading to a complete Cholesky factorization. A choice of ψ=1 will cause all off-diagonal elements to be rejected. Ajiz and Jennings [Ref. 13: p. 952] recommend that the rejection parameter be in the range $0.01<\psi<0.2$ for effective incomplete factorizations.

The diagonal modifications in C which result from the rejection of an element e^*_{ij} are given by

$$c_{jj}^{(k)} = |e^*_{kj}|(\bar{e}_{jj}/\bar{e}_{kk})^{1/2} \quad , \tag{3.21}$$

and by

$$c_{ii}^{(k)} = |e^*_{ik}|(\bar{e}_{ii}/\bar{e}_{kk})^{1/2} \quad . \tag{3.22}$$

With the successive application of equations 3.14 in conjunction with the rejection operation, 3.17, 3.16 and 3.15, all elements in column j of L are determined.

2. *The Conjugate Gradient Iteration*

The conjugate gradient method of Hestenes and Stiefel [Ref. 18] is applied to matrix equation 3.10. It uses the following vectors, the letter k indicates the k-th iteration,

a) $p^{(k)}$ conjugate gradient vector
b) $r^{(k)}$ residual vector

c) $y^{(k)}$ solution vector

d) $u^{(k)}$ product of $L^{-1}BB^{T}L^{-T}$ and $p^{(k)}$.

The initial values for k=0 are $y^{(0)} = 0$ and $p^{(0)} = r^{(0)} = b$. The algorithm for one iteration is as follows

$$u^{(k)} = L^{-1}BB^{T}L^{-T}p^{(k)}$$

$$\alpha_k = (r^{(k)})^{T}r^{(k)}/(p^{(k)})^{T}u^{(k)}$$

$$y^{(k+1)} = y^{(k)} + \alpha_k p^{(k)}$$

$$r^{(k+1)} = r^{(k)} - \alpha_k u^{(k)}$$

$$\beta_k = (r^{(k+1)})^{T}r^{(k+1)}/(r^{(k)})^{T}r^{(k)}$$

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)} .$$

The first step in the above algorithm is obtained without computing the transformed matrix explicitly by the following three operations,

1. $L^{T}v^{(k)} = p^{(k)}$ (back substitution)
2. $w^{(k)} = BB^{T}v^{(k)}$ (pre-multiplication)
3. $Lu^{(k)} = w^{(k)}$ (forward substitution).

The algorithm is terminated when

$$\|r^{(k)}\| / \|\bar{b}\| < \text{tolerance} \qquad (3.23)$$

where b equals the starting residual $r^{(0)}$, since $y^{(0)}$ was chosen to be zero.

IV. MODIFICATIONS AND COMPUTATIONAL RESULTS

A. ALGORITHMIC MODIFICATIONS

1. *Iterative Improvements*

A provision to improve the solutions to equations 2.21 and 2.22 has been implemented, because their residuals are often unduly high. The residuals are computed as

$$r = B_1 B_1^T \lambda - b' , \tag{4.1}$$

where λ stands for λ_0, λ_1 and λ_2 respectively and b' is generic for $B_1^T c_1'$, $(B_2)_1$ and $(B_2)_2$.

If $|r_i| > \varepsilon$ for some i, where ε is set to, say, 10^{-6}, the following systems are solved for λ'

$$B_1 B_1^T \lambda' = -r. \tag{4.2}$$

An improved solution is then obtained by adding λ and λ'

$$B_1 B_1^T (\lambda + \lambda') = r + b' + (-r) = b'. \tag{4.3}$$

The improvement can be repeated if the residuals of equation 4.3 are still found to be too high. With this modification the direct method (MDOC) has become a semi-iterative method.

2. *Removal of the Artificial Column*

The update of the artificial column with the residuals of the current solution

$$A(x', x'_{n+1}) - b x'_{n+2} = r \tag{4.4}$$

$$Ax^{(k)} - b = Ax^{(k-1)} - b - r / x'^{(k)}_{n+2} \tag{4.5}$$

on each iteration has been augmented, such that the artificial variable can be deleted from the problem when the infeasibility becomes small. After the deletion, only one dense column remains in B, and the solution of the least-squares problem is simplified. The motivation for this modification is to avoid some of the numerical problems that arise from variables that approach zero.

3. *Positive Semidefinite BB^T*

It can be shown that if, and only if a diagonal element ever goes to zero in a Cholesky factorization, BB^T is positive semidefinite, not positive definite. Furthermore, all elements below the zero diagonal element must also be zero and a nonunique solution to the normal equations can be obtained by setting the corresponding λ_i to 0. This solution can be obtained by replacing the zero diagonal element with any positive value and continuing with the factorization. Restated, if a diagonal element l_{jj} in the partially computed Cholesky factor is less than or equal to 10^{-6}, set $l_{jj}=1$, and $l_{ij}=0$ for $i=j+1,j+2,\ldots,m$ in equations 3.2 and 3.3 respectively. This procedure is also useful to deal with numerical problems which arise from degeneracy near optimality.

4. *Weighted Homogeneity Variable*

The initial solution $(y^o,y^o_{n+1},y^o_{n+2}) = 1$ used to begin the projective algorithm is arbitrary, and in some sense, the "homogeneity variable" y^o_{n+2} is fundamentally different than the other variables. Consequently, a modification has been made to allow weighting the starting solution y^o_{n+2} differently from the other y^o_i, $i=1,\ldots,n+1$. Let $y^o_{n+2}=s$. With $1^Ty^o=n+2$, the other y^o_i are set equal,

$$y^o_i = (n+2-s)/(n+1), \quad i=1,\ldots,n+1. \tag{4.6}$$

It is hoped that such a weighting might lead to lower iteration counts.

B. TEST PROBLEMS AND COMPUTATIONAL RESULTS

At first, a small test problem (TEST1) was used to verify the correctness of the source code. The two algorithms were then tested on seven problems which have been also used for testing by Lustig [Ref. 4: p. 19]. Table 5 shows some of the characteristics of the test problems.

TABLE 5
TEST PROBLEMS

Name	*Rows*	*Columns*	*Logicals*	*Nonzeros*	*Density*
TEST1	6	2	6	10	0.833
AFIRO	27	32	19	95	0.103
ADLITTLE	56	97	41	522	0.096
SHARE2B	96	79	83	901	0.119
ISRAEL	174	142	174	2529	0.102
BRANDY	193	249	54	2204	0.046
E226	223	282	190	2578	0.041
BANDM	305	472	0	2659	0.018

Table 6 summarizes the computational results. All CPU times represent the time in seconds to set up the major part of the data structure, solve the LP, and write out a few parameters on each iteration and the solution. Times to read in the data from MPS format are not included.

The convergence criterion equation 2.31 is used with t=0.05 for all test problems but TEST1 and SHARE2B, where t=0.001 and t=0.01 respectively. No solutions have been obtained for the problems AFIRO, BRANDY and BANDM: the algorithm will not converge to the optimum.

The feasibility parameter ρ (equation 2.7) is set to 0.9995, giving the best overall performance of the algorithm. Tests with ρ=0.9999 and

ρ=0.99999 on the data sets TEST1 and SHARE2B, indicate that with these higher values the number of iterations necessary to attain feasibility is reduced, but the total number of iterations remains about the same. In addition, the accuracy of the solutions deteriorates.

An intermediate solution is declared feasible, i.e. the artificial variable is removed from the problem, as soon as $x_{n+1}M$ is less than 10^{-1}. The choice of 10^{-1} is arbitrary, and depends on the value of the optimal solution. Using alternate values of 1.0 or 10^{-2} makes little difference in the number of iterations necessary to attain feasibility.

TABLE 6
COMPUTATIONAL RESULTS

	MDOC			*ICCG*		
Problem	*Iterations*		*Time*	*Iterations*		*Time*
	feasible	*optimal*		*feasible*	*optimal*	
TEST1	1	6	0.03	1	6	0.04
ADLITTLE	13	21	1.96	13	21	2.18
SHARE2B	6	27	5.04	7	37	5.31
ISRAEL	19	98	311	19	89	347
E226	8	49	107	9	49	130

Factorization failures plagued both algorithms before the modification to accomodate a semidefinite BB^T was implemented. These failures occured near the optimum, e.g. E226, or with the ICCG algorithm when setting the rejection parameter to a value greater than zero. After implementing the semidefinite modification these factorization problems have been cured, but the ICCG algorithm now shows very slow convergence. For example, a solution to SHARE2B is obtained only after 112.36 CPU seconds with ψ=0.015 and all other parameters unchanged. Thus, because storage is not at a premium, only complete factorizations

are used. Satisfactory numerical results with the incomplete factorization are obtained only with the trivial test problem TEST1; computation times are always inferior.

The minimum-degree ordering works very well in practice. Computational cost seems to be moderate, e.g., 0.25 seconds for SHARE2B, and 3.12 seconds for BANDM. Table 7 gives densities of BB^T and the Cholesky factors with and without reordering. Substantial reductions in storage requirements were achieved when doing the reordering.

TABLE 7
DENSITIES

			Re-ordered		*Not re-ordered*	
	Nonzeros		*Nonzeros*		*Nonzeros*	
Name	*in* BB^T	*Density*	*in L*	*Density*	*in L*	*Density*
TEST1	20	0.952	20	0.952	20	0.952
AFIRO	62	0.177	107	0.305	194	0.553
ADLITTLE	384	0.241	411	0.258	816	0.512
SHARE2B	871	0.187	1021	0.219	1134	0.243
ISRAEL	11227	0.737	11433	0.751	13743	0.903
BRANDY	2853	0.152	3429	0.183	9760	0.521
E226	2823	0.113	3639	0.146	10735	0.430
BANDM	3724	0.080	4660	0.100	32090	0.688

The densities and number of nonzeros in L or BB^T are relative to a symmetric half of a matrix; the diagonal elements are not included.

Computational results indicate that iterative improvements are not always an absolute necessity. Residuals of a current solution tend to be high at the start of iterations, probably due to a less than optimal

choice of M. Doing a few (2 or 3) iterative improvements at this stage stabilizes the computations until the algorithm gets close to the optimal solution. Numerical problems near the optimum are so severe that even allowing a prohibitively high number like 25 iterative improvements has no apparent influence on the quality of the solution.

The mildest form of numerical difficulty near the optimum is slow convergence, e.g. ISRAEL and E226. For SHARE2B, a high quality solution is obtained with no numerical difficulty. This data set is chosen to test the weighted homogeneity variable modification. S is given several values in the range 1 to 100. Iteration counts range from 25 to 37. The quality of the solutions is sometimes reduced, however. Only s=35 (27 iterations) and s=50 (28 iterations) give high quality solutions with low iteration counts. Values of s greater than 75 creat convergence failures.

C. CONCLUSIONS

The low iteration counts for some of the test problems are promising, although CPU times seem to tell the difference. These high CPU times result from test problems having slow convergence near the optimum, which is believed to be due to many variables going to zero. Thus, a technique to drop variables going to zero from the LP could well speed up convergence.

The ICCG algorithm does not perform very well on the test problems. Computational results are generally inferior to those obtained with the MDOC algorithm. Thus, no further research into this method seems warranted.

In view of the numerical problems encountered when solving the least-squares problem with the normal equations approach and Cholesky factorization, another method that does not use square roots should be considered for implementation. A very promising candidate in this respect is Givens rotation. See Gentleman [Ref. 19] and George and Heath [Ref. 20].

LIST OF REFERENCES

1. Bell Laboratories Technical Memorandum 11216-840126-04, *A New Polynomial-Time Algorithm for Linear Programming*, by N.K. Karmarkar, 26 January 1984.

2. Karmarkar, N.K., "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, v. 4, no. 4, pp. 373-395, 1984.

3. Klee, V., and Minty, G. L., "How Good is the Simplex Algorithm ?," *Inequalities III*, Shisha, O., (editor), pp. 159-179, Academic Press, 1972.

4. Stanford University Technical Report SOL 85-5, *A Practical Approach to Karmarkar's Algorithm*, by I.J. Lustig, June 1985.

5. University of California, Davis, Graduate School of Administration, *A Reduced Gradient Variant of Karmarkar's Algorithm*, by D.F. Shanno, March 1985.

6. Wood, R.K., Personal communication with Bretschneider, 7 December 1984.

7. Heath, M.T., "Numerical Methods for Large Sparse Linear Least Squares Problems," *SIAM Journal on Scientific and Statistical Computing*, v. 5, no. 3, pp. 497-513, 1984.

8. Heath, M.T., "Some Extensions of an Algorithm for Sparse Linear Least Squares Problems," *SIAM Journal on Scientific and Statistical Computing*, v. 3, no. 2, pp. 223-237, 1982.

9. Bjoerck, A., and Duff, I.S., "A Direct Method for the Solution of Sparse Linear Least Squares Problems," *Linear Algebra and its Applications*, v. 34, pp. 43-67, 1980.

10. Brameller, A., Allan, R.N., and Hamam, Y.M., *Sparsity*, Pitman, 1976.

11. George, A., Heath, M.T., and Ng, E., "A Comparison of some Methods for Solving Sparse Linear Least-Squares Problems," *SIAM Journal on Scientific and Statistical Computing*, v. 4, no. 2, pp.177-187, 1983.

12. George, A. and Liu, J.W.-H., *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, 1981.

13. Ajiz, M.A., and Jennings, A., "A Robust Incomplete Choleski-Conjugate Gradient Algorithm," *International Journal for Numerical Methods in Engineering*, v. 20, pp. 949-966, 1984.

14. Jennings, A., and Malik, G.M., "Partial Elimination," *Journal of the Institute of Mathematics and its Applications*, v. 20, pp. 307-316, 1977.

15. Jennings, A., and Malik, G.M., "The Solution of Sparse Linear Equations by the Conjugate Gradient Method," *International Journal for Numerical Methods in Engineering*, v. 12, pp. 141-158, 1978.

16. Golub, G.H., and Van Loan, C.F., *Advanced Matrix Computations*, John Hopkins, 1983.

17. Jennings, A., and Ajiz, M.A., "Incomplete Methods for Solving $A^TAx=b$," *SIAM Journal on Scientific and Statistical Computing*, v. 5, no. 4, pp. 978-987, 1984.

18. Hestenes, M.R., and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, v. 49, pp. 409-436, 1952.

19. Gentleman, W.M., "Least Squares Computations by Givens Transformations Without Square Roots," *Journal of the Institute of Mathematics and its Applications*, v. 12, pp. 329-336, 1973.

20. George, A., and Heath, M.T., "Solution of Sparse Linear Least Squares Problems Using Givens Rotations," *Linear Algebra and its Applications*, v. 34, pp. 69-83, 1980.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93943-5100	1
4.	Professor R.Kevin Wood Code 55Wd Department of Operations Research Naval Postgraduate School Monterey, California 93943-5100	14
5.	Professor Gerald G. Brown Code 55Bw Department of Operations Research Naval Postgraduate School Monterey, California 93943-5100	1
6.	Professor Gordon H. Bradley Code 52Bz Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	1
7.	Professor Arthur L. Schoenstadt Code 53Zh Department of Mathematics Naval Postgraduate School Monterey, California 93943-5100	1
8.	Professor Richard E. Rosenthal Code 55Ro Department of Operations Research Naval Postgraduate School Monterey, California 93943-5100	1
9.	Professor Richard McBride FBE Department University of Southern California Los Angeles, California 90089-1421	1
10.	Professor Glenn W. Graves Graduate School of Management University of California, Los Angeles Los Angeles, CA 90024	1
11.	Captain Guenter W. Bretschneider Materialamt der Luftwaffe Postfach 902500 5000 Koeln 90, West Germany	5

216071

Thesis
B80432
c.1

Bretschneider
An implementation of the projective algorithm for linear programming.